VŠB – Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Vývoj aplikací na prototypové platformě .NET Gadgeteer

.NET Gadgeteer Application Development

Student: Bc. Karla Sladká
Vedoucí diplomové práce: Ing. Marek Babiuch, Ph.D.

Ostrava 2015

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání diplomové práce

Student: **Bc. Karla Sladká**

Studijní program: N2301 Strojní inženýrství

Studijní obor: 3902T004 Automatické řízení a inženýrská informatika

Téma: Vývoj aplikací na prototypové platformě .NET Gadgeteer
.NET Gadgeteer Application Development

Zásady pro vypracování:

1. Popište platformu .NET Gadgeteer a její hardwarové prostředky, vývojové desky a Gadgeteer moduly.
2. Proveďte rozbor vývoje aplikací, vývojové nástroje a softwarovou podporu.
3. Sestavte vývojovou sadu FEZ Cerbot a naprogramujte vlastní úlohu s využitím Gadgeteer senzorů.
4. Pro vybranou vývojovou desku vytvořte úlohu využívající síťovou komunikaci.
5. Navrhněte, implementujte a zdokumentujte sadu laboratorních úloh využívající vývojové desky a senzorické Gadgeteer moduly.

Seznam doporučené odborné literatury:

D. Thompson, and R. Miles, "4. Embedded Programming with the Microsoft® .NET Micro Framework," Microsoft Press, ISBN 978-0735623651, 2007.
Sytech Designs web portal, available at: http://www.sytechdesigns. com/micro_framework.htm, 2014.
Microsoft, Micro Framework websites, available at: http://www.netmf.com/, 2014.
GHI Electronics, "GHI Tutorials", available at:
https://www.ghielectronics.com/docs/37/netmf-and-gadgeteer-tutorial-index, 20144.
J. Kuhner, "5. Expert .NET Micro Framework (Expert's Voice in .NET)," Springer-Verlag New York, ISBN 978-1-59059-973-0, 2008.
S. Monk, "Getting Started with .NET Gadgeteer," O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, May 2012, ISBN 978-1-449-32823-8.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Marek Babiuch, Ph.D.**

Datum zadání: 13.12.2014

Datum odevzdání: 18.05.2015

doc. Ing. Renata Wagnerová, Ph.D.
vedoucí katedry

doc. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Poděkování

Ráda bych poděkovala vedoucímu práce Ing. Marku Babiuchovi, Ph.D. za poskytnutí tématu, odborné vedení, cenné rady a pomoc při vypracovávání diplomové práce.

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě dne: .....18.5.2015.....

.....

podpis studenta

Prohlašuji, že

- jsem byl seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě diplomovou (bakalářskou) práci užít (§ 35 odst. 3).
- souhlasím s tím, že diplomová (bakalářská) práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou (bakalářskou) práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě dne: .....18.5.2015.....

.....

podpis

Jméno a příjmení autora práce: Karla Sladká

Adresa trvalého pobytu autora práce: J. Seiferta 1, 74801 Hlučín

.

ANOTACE

Tato práce představuje platformu .NET Gadgeteer, která úzce spojuje hardwarové elektronické komponenty se softwarem, který je jejich součástí. Technologie .NET Gadgeteer vychází z technologie .NET Micro Framework a je propojená s technologií elektronických komponentů FEZ. Práce popisuje architekturu .NET Micro Framework a specifikaci jednotlivých vývojových desek, senzorických a komunikačních modulů. Praktická část práce se zabývá tvorbou aplikací postavených na platformě .NET Gadgeteer. První oblast aplikuje použití senzorů, následně se práce zaměřuje na síťovou komunikaci, jak mezi zařízeními platformy .NET Gadgeteer, tak na připojení k osobnímu počítači s využitím Ethernet nebo WiFi. V závěru práce je implementována aplikace z oblasti robotiky.

Klíčová slova: .NET, .NET Micro Framework, Gadgeteer, FEZ, Visual Studio, C#

ABSTRACT

Theses introduce .NET Gadgeteer platform. This platform connects hardware components and software inside the components. Gadgeteer is built on technologies .NET Micro Framework and is connected with technology of electronics devices FEZ. Theses describes architecture of .NET Micro Framework also and introduces some mainboard with external modules. Practical part of theses is dealing with the implementation of applications. Applications are built on .NET Gadgeteer platform. First part of applications is used sensors. Next chapter theses is focused on networking between devices of platform .NET Gadgeteer or connect to PC using Ethernet or WiFi. Last chapter is devoted to implemented application of robotic area.

Keywords: .NET, .NET Micro Framework, Gadgeteer, FEZ, Visual Studio, C#

Obsah

SEZNAM POUŽITÝCH SYMBOLŮ

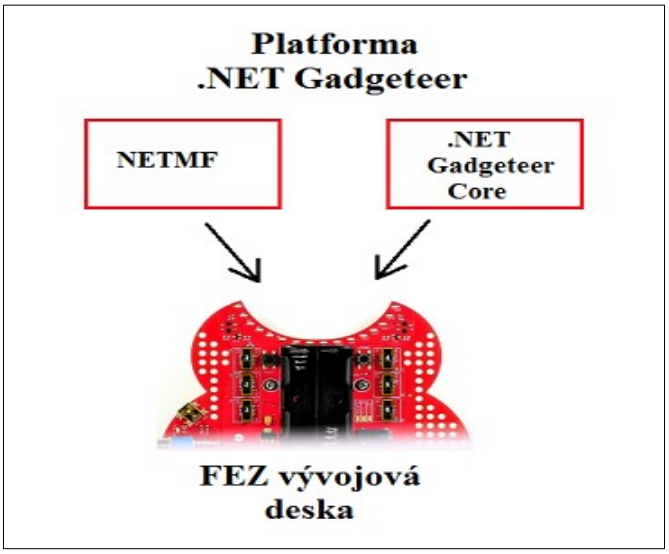
.NET	Název souboru technologií firmy Microsoft
Ad hoc	Dočasné síťové připojení mezi dvěma prvky sítě
C#	Programovací jazyk
CAN	(Controller Area Network) sběrnice
CLR	(Common language runtime) Správa mezi softwarem a hardwarem
DHCP	(Dynamic Host Configuration Protocol) server ke konfiguraci sítě
DLL	Dynamicky linkovaná knihovna
FEZ	(Freaking Easy) technologie pro zjednodušení elektronických součástek
GSM	(Groupe Spécial Mobile) Globální Systém pro Mobilní komunikaci
HAL	(Hardware abstraction layer) Abstraktní hardwarová vrstva
Hz	(Hertz) jednotka frekvence
HW	Hardware
I2C	(Inter-Integrated Circuit) počítačová sériová sběrnice
LED	(Light-Emitting Diode) dioda emitující světlo
NETMF	.NET Micro Framework
PAL	(platform abstraction layer) platforma mezi ovladači a runtime vrstvou
PWM	(Pulse Width Modulation) pulzně šířková modulace
RAM	(Random Access Memory) druh paměti
RSSI	(Received Signal Strength Indicator) Indikátor síly příjmaného signálu
SD	(Secure Digital) paměťová karta
SSID	(Service Set Identifier) identifikátor každé bezdrátové počítačové sítě
SPI	(Serial Peripheral Interface) sériové periferní rozhraní
TCP/IP	(Transmission Contro lProtocol/Internet Protocol) přenosový protokol
USART	(Universal Synchronous Receiver and Transmitter) sériové rozhraní
USB	(Universal Serial Bus) je univerzální sériová sběrnice

WEP	(Wired Equivalent Privacy) druh kódování bezdrátových sítí
WiFi	Standard bezdrátové sítě
WPA2	(WiFi protected access) druh kódování bezdrátových sítí
SPOT	(Smart Personal Object Technology) chytrá objektová technologie

1 .NET GADGETEER A SOUČASNOST

.NET Gadgeteer vychází z pojmu Gadgeteering. Tento pojem označuje blízké spojení hardwaru se softwarem. Tento projekt vznikl z potřeby vytvořit elektronické součástky pro jednoduchou práci a manipulaci, zejména pro vzdělávací účely. Projekt .NET Gadgeteer vznikl za podpory firmy Microsoft převážně u tvorby vývojového prostředí a knihoven programového kódu. Podpora vývoje kódu je na základu .NET Micro Framework (NETMF), ke kterému jsou vytvořené grafické knihovny, ovladače a další softwarová podpora pro vývoj aplikací platformy Gadgeteer. Knihovny s NETMF mají volně přístupný kód a jsou tedy vnímány jako open-source. Technologie umožňuje rozšířit hardware o zařízení, či senzory, které nejsou součástí modulů Gadgeteer a vytvořit k nim vlastní knihovny, na základu NETMF. Výhodou se stává právě široká podpora zdrojového kódu jazyka C# pro tvorbu aplikací, které jsou pak snadno aplikovatelné na vývojovou desku a moduly. K modulům a deskám se navíc přistupuje objektově, což posouvá vývoj aplikací do vyšší úrovně i u práce s HW komponenty. Vysoká kompatibilita s webovými a desktopovými aplikacemi, práce s webovými službami a možnost využití na mobilních zařízeních. Vzhledem k nízkým nárokům na HW a vysokoúrovňového jazyka platformy nelze pracovat se systémem v reálném čase.

Tato technologie se dostala na trh okolo roku 2009 a již má zastoupení v široké komunitě a na fórech vývojářů. Její výhodou se stává možnost tvorby vlastních knihoven a aplikací, popřípadě připojení vlastních zařízení pomocí komunikačních modulů, či standardizovaného rozhraní.



Obr. 1 Platforma .NET Gadgeteer

2 TECHNOLOGIE SPOJENÉ S PLATFORMOU .NET GADGETEER

Platforma .NET Gadgeteer se skládá z technologií .NET Micro Framework a .NET Gadgeteer knihoven ve spojení s technologií hardwarových komponentů FEZ.

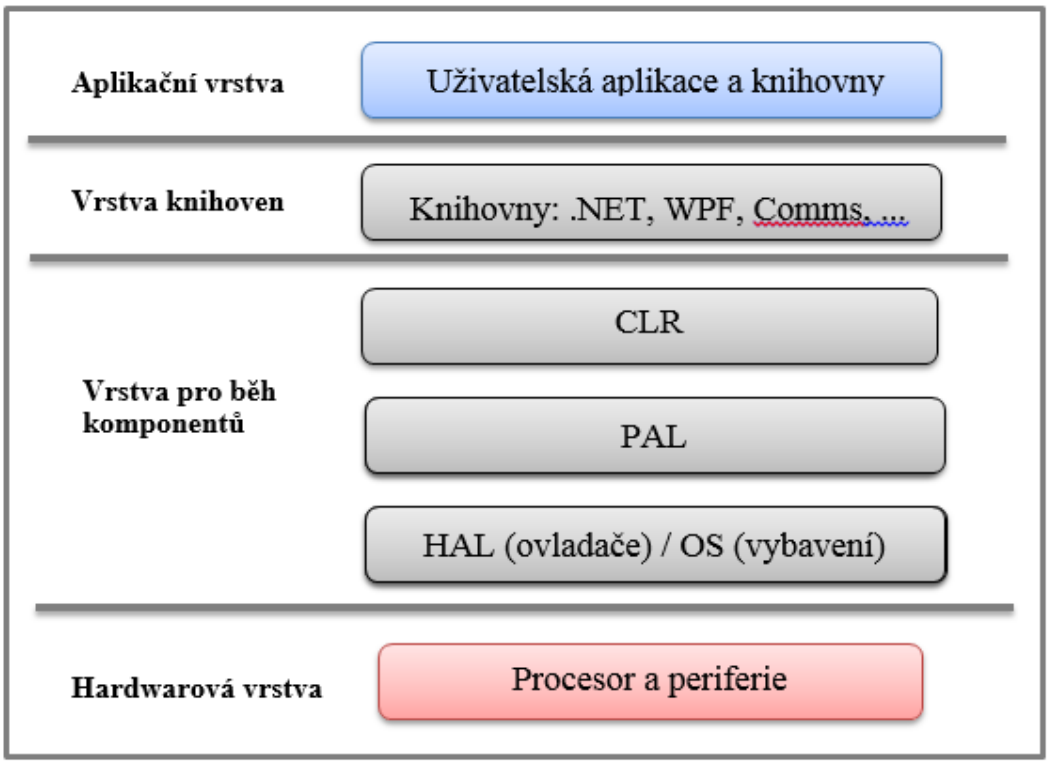
2.1 .NET Micro Framework

.NET technologie, která vznikla pro vytváření programů na omezených zařízeních, jako jsou menší 32 bitové mikroprocesory pouze s několika kilobitů RAM paměti a paměti programu. .NET Micro Framework (dále jen NETMF) je malá a efektivní platforma, využívaná pro vývoj aplikací prostředí Visual Studia a programovací jazyky C#, či VB. K vývoji programů pro mikroprocesor, lze používat ty samé nástroje a jazyky jako při vytváření aplikací pro desktopové nebo pro chytré mobilní zařízení. Aplikace založené na technologii NETMF mohou pracovat se zařízením pomocí komponent PAL a HAL, které se do zařízení importují. Poté lze na HW zařízení splňující minimální požadavky pro platformu nahrát aplikaci vytvořenou v NETMF. Další součástí je HW emulátor pro rychlé ladění a nahrání programu do zařízení bez větších komplikací.

Architektura NETMF se skládá ze čtyřech vrstev viz obr. 2. První vrstva je vrstva Hardwarová, obsahuje mikroprocesor a další prvky elektronického obvodu. Další vrstvou je podmnožina komponentů umožňující běh aplikace na HW vrstvě tzv. RCL. Základní prvky vrstvy jsou komponenty CLR, PAL a HAL. CLR poskytuje aplikacím správu systému při práci s pamětí, vlákny, běžícími procesy atd. PAL a HAL jsou komponenty C++ funkcionality volané CLR komunikující s nižší HW vrstvou. Nadřazená vrstva Knihovna tříd obsahuje NETMF objektově orientované kolekce funkcí a objektů k vývoji aplikací v jazyce C# podporující šifrování, debatování, grafiku, základy knihoven DLL, CLR knihovny, knihovny obsahující komunikační standardy. Nejvyšší vrstva se nazývá aplikační vrstva a stará se o vytvořené aplikace nahraných do zařízení.

NETMF nevyžaduje jiné platformy, operační systém nebo jednotku správy paměti (MMU). K tomu má TinyCLR, což je zjednodušené běhové prostředí pro spouštění kódu a služeb usnadňující proces vývoje. Obsahuje ovladače pro běžné periferie a komunikační rozhraní například paměť Flash, EEPROM, GPIO, I^2C, USB, SPI, RS232, RS485 se vzrůstajícím

počtem komunikačních modulů roste i počet podporovaných rozhraní. NETMF běží bez nutnosti spotřebování větší množství energie.



Obr. 2 Architektura NETMF

Technologie NETMF byla vyvíjena od roku 2001 jako technologii SPOT. V roce 2006 byla představena první verze NETMF 1.0, která byla implementována na robotickém emulátoru zvané sumo. Další verze NETMF 2.0 byla vydána v roce 2007, kdy byly již vytvořeny HW desky, na kterých nová verze NETMF běžela. Další rozšiřující verze NETMF 2.5 byla vydána v únoru roku 2008. Dnes již máme nejnovější dostupnou verzi NETMF 4.3, která je dostupná od prosince 2013.

Platforma NETMF pracuje velmi rychle a je vhodný pro většinu aplikací. Vzhledem ke sníženým nárokům na HW, nelze čekat deterministické chování v reálném čase. To znamená, že NETMF není platforma, která pracuje s HW v reálném čase. Jeden z důvodů je, že časovač událostí se nevyvolává přesně v daných intervalech. Dále mohou vznikat prodlevy při požadavku na přerušení, a to například v řádu desítek milisekund, než se přerušení vyvolá.

2.2 Technologie FEZ

HW komponenty .NET Gadgeteer jsou elektronické součástky postavené na technologii FEZ. Základní myšlenou technologie se stává vytvoření elektronických součástek bez nutnosti pájení. Takové součástky, které lze jednoduše a rychle sestavit. Technologie FEZ si zakládá na tom, aby byly komponenty stabilní a kompatibilní.

Vývoj technologie FEZ byl spjatý s technologií NETMF a jejími SW komponenty TinyCRL pro běh aplikací. Další myšlenkou je umožnit rychlý a snadný vývoj aplikací. To znamená, že byly vytvořeny ovladače pro každý HW technologie FEZ, které se nachází v SDK NETMF a jsou součástí komponenty HAL. Ovladače mají za úkol zajistit lepší kompatibilitu komponentů FEZ se softwarem, zejména u aplikování vytvořeného programu na HW. Nejprve vznikly vývojové desky, které obsahovaly mikroprocesor, paměti, komunikační rozhraní, napájení, tlačítka a jiné.

Prvním produktem technologie FEZ bylo FEZ Domino vydané v roce 2010. TinyCRL vytvářela komunitu, jakmile na trh přišly vývojové desky FEZ Mini s FEZ Cobra. Komunita se stále rozšiřovala s nově příchozími vývojovými deskami. V první polovině roku 2011 byla na trh uvedena platforma .NET Gadgeteer spolu s deskou FEZ Spider. Poté byla ještě v roce 2011 vydána open-source deska FEZ Hydra, která byla výkonnější než FEZ Spider.

3 VÝVOJOVÉ DESKY A MODULY .NET GADGETEER

Platforma .NET Gadgeteer je založena na rychlé a jednoduché spolupráci s jednotlivými hardwarovými komponenty, které lze jednoduše poskládat a následně i programovat bez nutnosti instalace programátoru nebo restartování zařízení. Celý projekt se skládá z vývojové desky zvané Mainboard, přídavných modulů a programového vybavení tvořeného z platformy.Net Micro Framework a knihoven, či grafického rozhraní od společnosti GHI Electronics.

3.1 Řídicí moduly

Moduly jsou základem jednotlivých typů vývojových desek. Obsahují procesor i externí krystal, integrované paměti, periferie, grafické rozhraní a knihovny.

3.1.1 EMX modul

Základem modulu je 32-bitový procesor ARM7, uživatelská paměť flash 2,539MB a paměť RAM 11MB. Součástí modulu jsou síťové protokoly Ethernet, TCP/IP, WiFi a SSL, grafické knihovny pro podporu obrazovek a USB host a klient. Modul tvoří jádro desky FEZ Spider, kde je v rámci desky rozšířen o sloty k externím modulům.

3.1.2 Cerb 40 II modul

Modul obsahuje 32-bitový procesor Cortex-M4 a krystal o frekvenci 168MHz. Součástí modulu jsou uživatelské paměti typu flash o velikosti 384KB a RAM s velikostí 104 KB. Modul má podporu síťových rozhraní u protokolu Ethernet a TCP/IP, debugger přes USB klient. Jádro desek FEZ Cerberus a FEZ Cerbot je založeno právě na modulu Cerb 40. [GHI Electronics]

3.2 Externí moduly

Technologie .NET Gadgeteer má rozmanitou škálu externích modulů, kterou postupně rozšiřuje, a tím dává větší možnosti pro vytváření elektronických projektů a jejich vývoje oblasti aplikací a hardwaru.

Externí moduly mají dané typy slotů, které se liší podle jednotlivých typů modulů a jejich technologií. Pro napájení modulů se používá napětí 3,3V nebo 5V a 35mA.

[GHI Electronics]

Socket	Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8	Pin9	Pin10
A	+3.3V	+5V	AIN (G!)	AIN (G)	AIN	GPIO	[UN]	[UN]	[UN]	GND
B	+3.3V	+5V	LCD B0	LCD B1	LCD B2	LCD B3	LCD B4	LCD ENABLE	LCD CLK	GND
C	+3.3V	+5V	GPIO!	CAN TD (G)	CAN RD (G)	GPIO	[UN]	[UN]	[UN]	GND
D	+3.3V	+5V	GPIO!	D-	D+	GPIO	GPIO	[UN]	[UN]	GND
E	+3.3V	+5V	[UN]	LED1 (OPT)	LED2 (OPT)	TX D-	TX D+	RX D-	RX D+	GND
F	+3.3V	+5V	GPIO!	DAT0	DAT1	CMD	DAT2	DAT3	CLK	GND
G	+3.3V	+5V	LCD G0	LCD G1	LCD G2	LCD G3	LCD G4	LCD G5	LCD BACKLIGHT	GND
H	+3.3V	+5V	GPIO!	D-	D+	[UN]	[UN]	[UN]	[UN]	GND
I	+3.3V	+5V	GPIO!	[UN]	[UN]	GPIO	[UN]	SDA	SCL	GND
K	+3.3V	+5V	GPIO!	TX (G)	RX (G)	RTS	CTS	[UN]	[UN]	GND
O	+3.3V	+5V	GPIO!	GPIO	AOUT	[UN]	[UN]	[UN]	[UN]	GND
P	+3.3V	+5V	GPIO!	[UN]	[UN]	GPIO	PWM (G)	PWM (G)	PWM	GND
R	+3.3V	+5V	LCD R0	LCD R1	LCD R2	LCD R3	LCD R4	LCD VSYNC	LCD HSYNC	GND
S	+3.3V	+5V	GPIO!	GPIO	GPIO	CS	MOSI	MISO	SCK	GND
T	+3.3V	+5V	[UN]	YU	XL	YD	XR	[UN]	[UN]	GND
U	+3.3V	+5V	GPIO!	TX (G)	RX (G)	GPIO	[UN]	[UN]	[UN]	GND
X	+3.3V	+5V	GPIO!	GPIO	GPIO	[UN]	[UN]	[UN]	[UN]	GND
Y	+3.3V	+5V	GPIO!	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GND
Z	+3.3V	+5V	[MS]	[MS]	[MS]	[MS]	[MS]	[MS]	[MS]	GND
*	+3.3V	+5V	GPIO!	GPIO	GPIO	[MS]	[MS]	[MS]	[MS]	GND

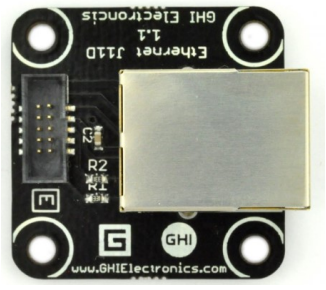
Obr. 3.1 Typy slotů[GHI Electronics]

Jednotlivé typy externích modulů můžeme rozdělit do kategorií:

- **Audio**
    - Vytváří potvrzovací zvuky nebo nastavené melodie
- **Kamery**
    - Různé typy kamer s odlišnými funkčními vlastnostmi a rozlišením, které je dáno zejména velikosti a cenou kamery.

Obr. 3.2 Modul kamery[GHI Electronics]

- **Ovladače**
    - Konektory, ovladače pro motor, registry pro řetězení modulů, které umožňují rozšířit možnosti vytváření projektů například řízení vlastních HW zařízením.
- **Ethernet**
    - Ethernetové moduly typu J11D a ENC28 k síťovému propojení mezi moduly nebo počítačem.



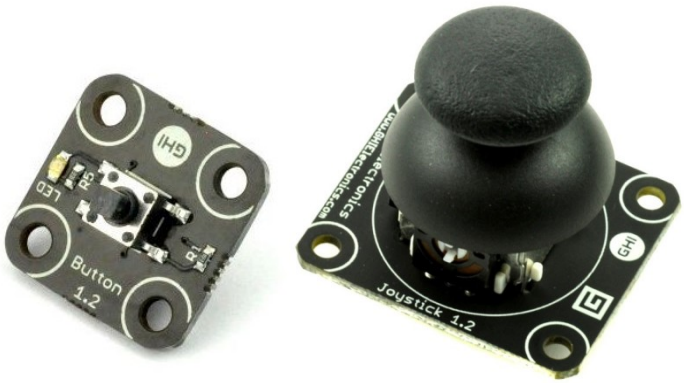
Obr. 3.3 Ethernet modul [GHI Electronics]

- **Rozhraní**
    - USB hostitelský modul, Hub modul, RS232 modul pro použití k připojení vlastních modulů

- **Napájení**
    - USB klient modul, napájecí externí modul, bateriový modul a další
- **Senzory**
    - Akcelerometr, kompas, gyroskop, senzor vlhkosti, senzor pro zjištění plynů, a další

Obr. 3.4 GasSense modul a LightSense modul [GHI Electronics]

- **Speciální moduly:**
    - Čtečky karet, G-plug pro bezdrátové připojení a jiné
- **Uživatelské vstupy**
    - Tlačítka, joystick, dotykové moduly, rotační čidlo a další



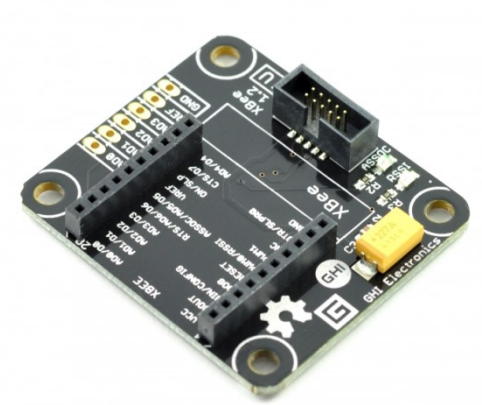
Obr. 3.5 Modul tlačítka a joysticku [GHI Electronics]

- **Paměťové moduly**
    - Modul pro paměťové SD karty mají za úkol dle potřeby rozšířit úložiště pro obrázky, videa nebo pro naměřená data.

- Zobrazovací **moduly**
    - K dispozici jsou zejména různé druhy obrazovek například znakové, barevné, dotykové a LED

Obr. 3.6 Modul dotykové obrazovky[GHI Electronics]

- **Bezdrátové moduly**
    - Bezdrátové připojení pro delší i kratší vzdálenosti k přenosu dat lze využít Wifi moduly, XBee modul, GSM radiový modul



Obr. 3.7 XBee modul[GHI Electronics]

3.3 Druhy vývojových desek

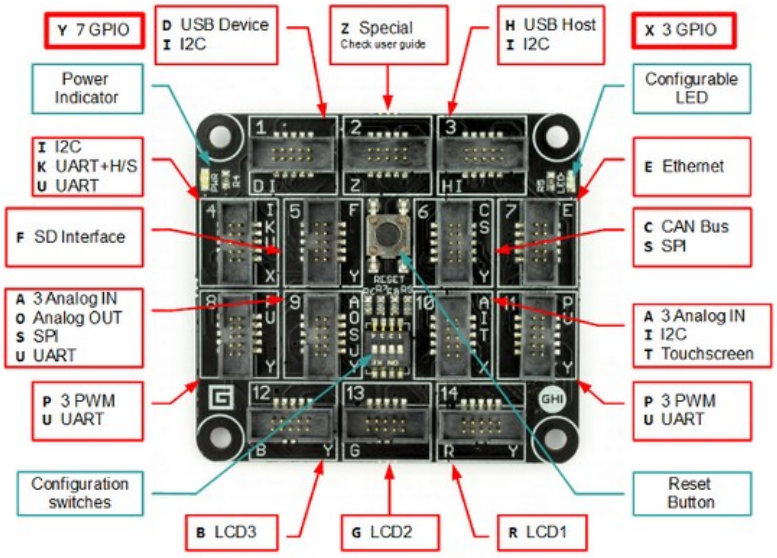
Na trhu je několik odlišných druhů řídicích desek platformy .NET Gadgeteer, které se liší podle zakomponovaných řídicích modulů. Dále se liší v počtu integrovaných patic a integrovaných rozšíření v podobě napájení z baterie, či integrovaných modulů pro bezdrátovou komunikaci.

3.3.1 FEZ Spider

Je to základní typ vývojové desky, kterou lze získat jako součástí základní startovací sady. Podporuje síťové rozhraní, souborové systémy a grafické rozhraní.[GHI Electronics]

Specifikace:

- Procesor 72MHz 32-bit ARM7
- Flash paměť 2,5 MB
- RAM 11 MB
- Počet slotů 14
- Komunikační rozhraní: SPI, I2C, CAN, UART, GPIO
- Rozměry 57 x 52 x 11,65 mm



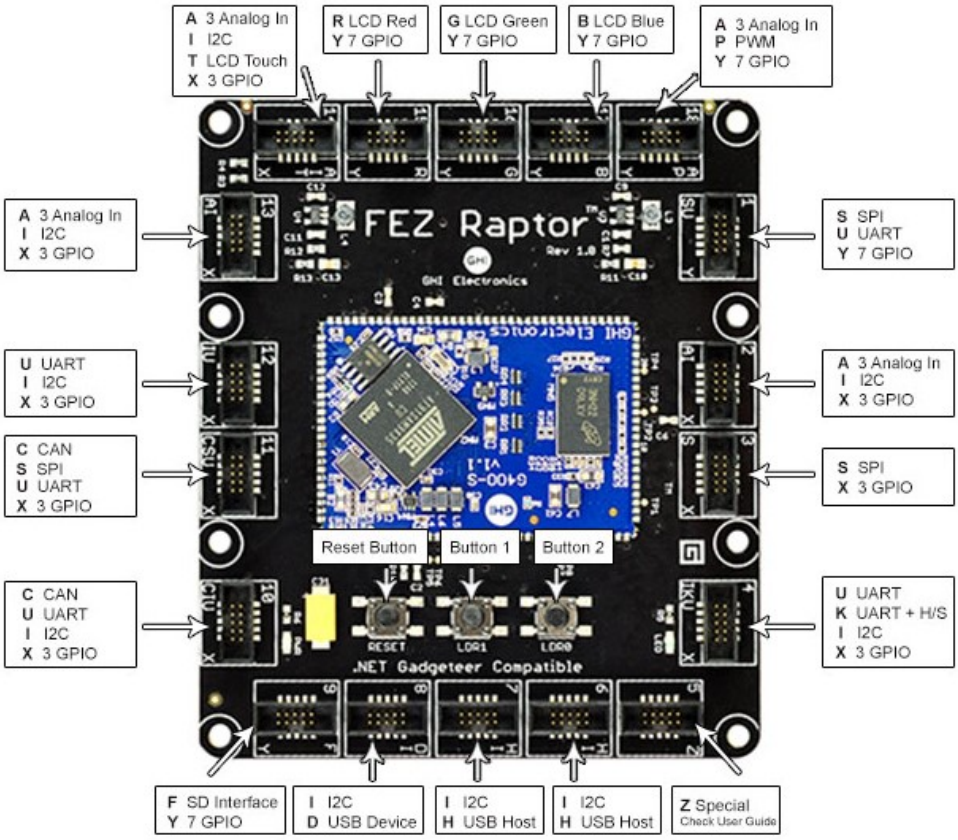
Obr. 3.8 FEZ Spider Mainboard [GHI Electronics]

3.3.2 FEZ Raptor

Nejvýkonnější vývojová deska na trhu. Podporuje síťové rozhraní, souborové systémy a grafické rozhraní. Navíc obsahuje 3 integrované tlačítka. [GHI Electronics]

Specifikace:

- Procesor 400MHz 32bit ARM9
- Flash paměť 1,4 MB
- RAM 92MB
- Počet slotů 18
- Komunikační rozhraní: SPI, I2C, CAN, UART,GPIO
- Rozměry 102 x 87 x 7,5 mm



Obr. 3.9 FEZ Raptor Mainboard [GHI Electronics]

3.3.3 FEZ Cerberus

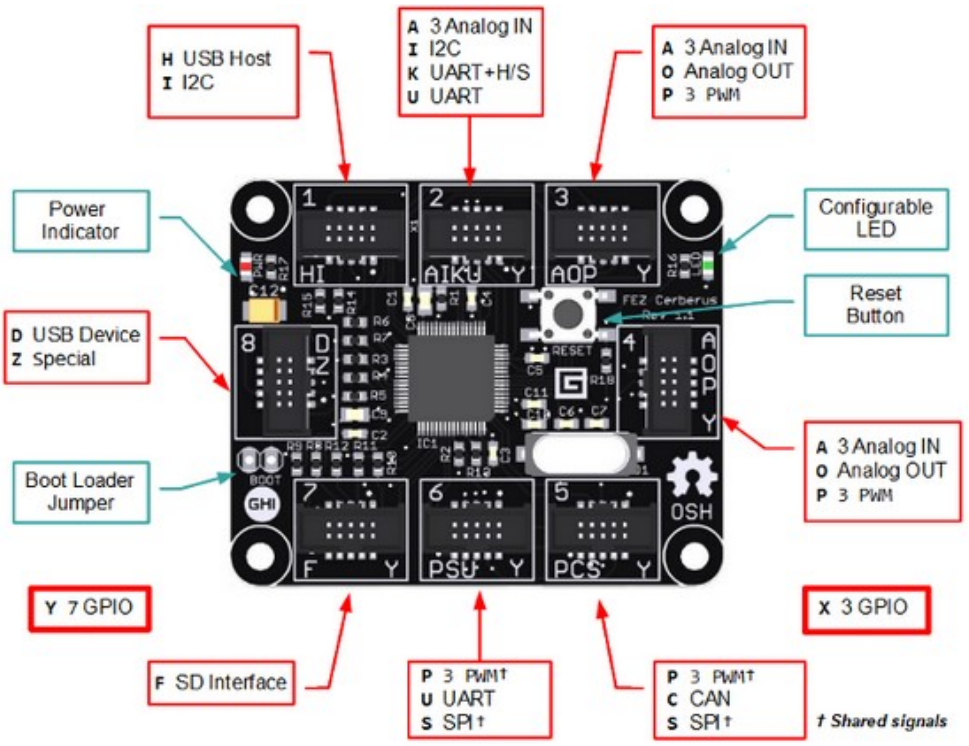
Jednoduchá a levná řídicí deska. Součástí jsou rozšířené knihovny, které umí pracovat se standardy síťového připojení, souborovými systémy, grafickým rozhraním a periferiemi.

[GHI Electronics]

Specifikace:

- Procesor 168MHz 32-bit Cortex-M4
- Flash paměť 384 KB
- RAM 119 KB
- Počet slotů 8
- Komunikační rozhraní: SPI, I2C, CAN, UART,GPIO
- Rozměry 57 x 47 x 7,2 mm

Exter



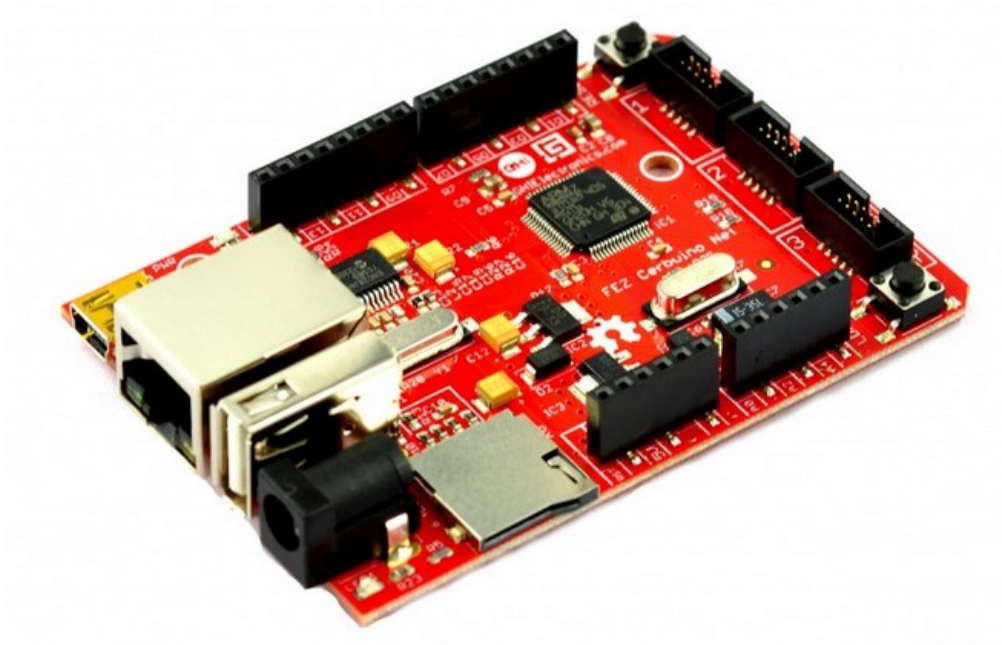
Obr. 3.10 FEZ Cerberus Mainboard [GHI Electronics]

3.3.4 FEZ Cerbuino Net

Vývojová deska obsahuje čipovou sadu z rodiny Cerberus, na které je založené celé jeho jádro. Deska má integrovaný modul pro rozhraní Ethernet s rozšířenými knihovnami u síťových protokolů TCP/IP, SSL sloužící k jednodušší tvorbě aplikace pro práci se síťovým rozhraním. Obsahuje slot na SD kartu, USB hosta a klienta. [GHI Electronics]

Specifikace:

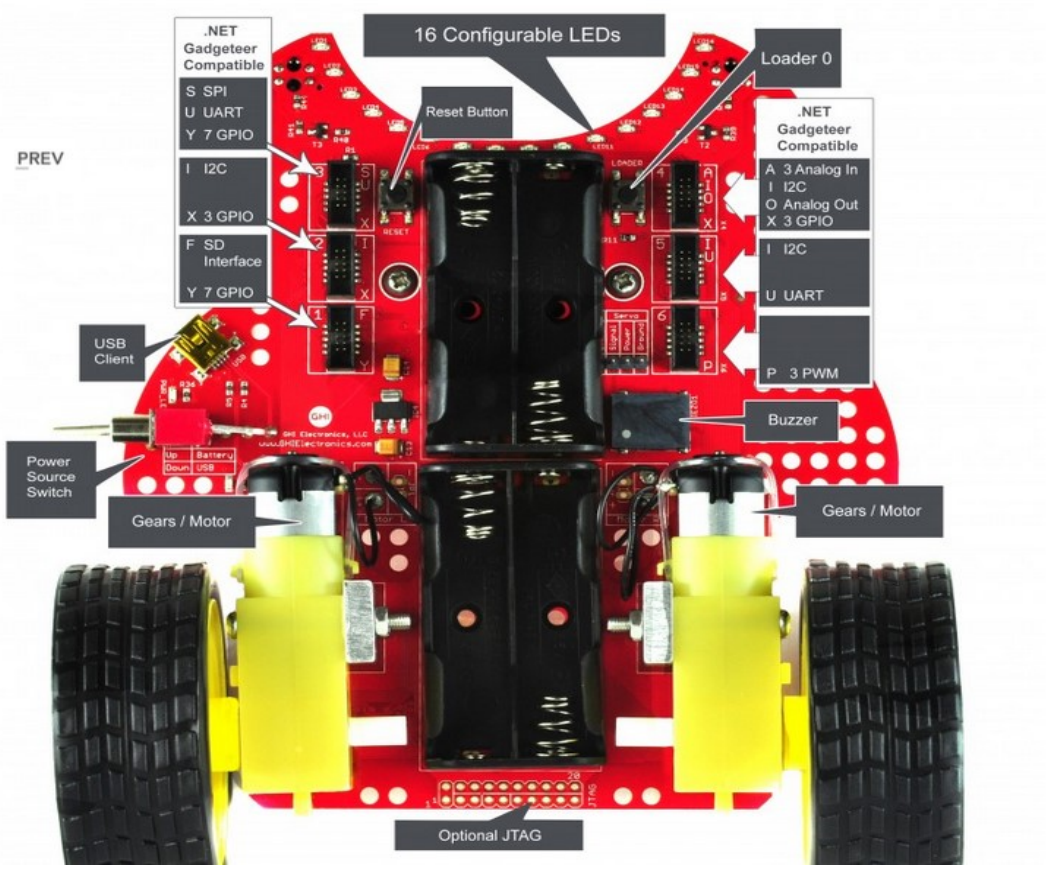
- Procesor 168MHz 32-bit Cortex-M4
- Flash paměť 384 KB
- RAM 119 KB
- RAM Ethernet 104KB
- Počet slotů 3
- Komunikační rozhraní: SPI, I2C, CAN, UART,GPIO
- Rozměry 102 x 87 x 7,5 mm



Obr. 3.11 FEZ Cerbuino Net [GHI Electronics]

3.3.5 FEZ Cerbot

FEZ Cerbot je spojením vývojové sady s deskou, která obsahuje integrované moduly pro sestavení pohyblivého vozítka. Jádro desky pochází z rodiny Cerberus, což znamená, že základ desky (procesor, RAM a Flash paměť,…) je stejný jako u FEZ Cerberus. Cerbot navíc obsahuje 16 ledek, zvukový generátor, dva motory s koly a řídicím čipem, přední kolečko a 4 držáky na baterie typu AA. Na desce jsou integrovány dva reflexní senzory umožňující vývojářům vytvořit aplikaci k sledování čáry s rozpoznání hran stolu a jiných okrajů. Díky 6 kompatibilním paticím lze Cerbot externími moduly rozšířit o kameru, bezdrátové komunikační moduly, externí slot Micro SD pro paměťovou kartu, display, akcelerometr gyroskop a další senzorické moduly. [GHI Electronics]



Obr. 3.12 FEZ Cerbot sestava [GHI Electronics]

4 MODULY PRO MĚŘICÍ A SENSOROVÉ APLIKACE

Pro vybrané Gadgeteer moduly a desky jsou vytvořeny ukázkové příklady pro využití přídavných modulů.

4.1 Aplikace s modulem potenciometru a LED světel

Příklad se zabývá vytvořením aplikace, která má rozsvítit daný počet světel na modulu s LED diodami podle hodnoty napětí na potenciometru.

4.1.1 Hardwarové komponenty aplikace

Součástí příkladu je deska FEZ Spider (popis desky se nachází v kapitole 3.3.1), ke které se připojí moduly, a to modul potenciometru, USB klient, modul LED7R obsahující 7 diod.

Jednotlivé moduly a jejich popis:

- Potenciometr
    - Využívá se pro měření pozice nebo ovládání úrovně.
    - Typ socketu A.
    - Napájení 3,3V s proudem < 1mA nebo 5V s proudem 0mA.
    - Velikost modulu je 27 x 28mm.

Obr. 4.1 Modul potenciometru[GHI Electronics]

- USB Client SP
    - Model slouží pro připojení USB Klienta k desce, která nemá USB jako integrovanou součást.
    - Slot pro připojení je typ D.
    - 3,3V a 5V.

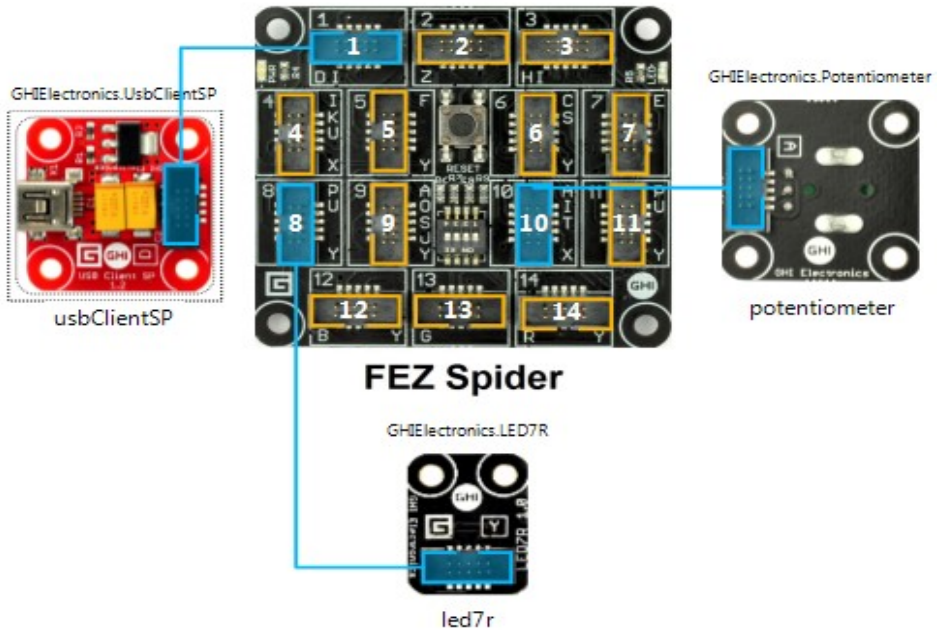
- Rozměry jsou 27 x 27mm.

- LED7R
  - Obsahuje 6 zelených led světel zapojených v kruhu a jednu červenou ve středu.
  - Pro připojení k desce je využit socket typu Y.
  - Napájení 3,3V a 35 mA nebo 5V s 0mA.
  - Rozměry jsou 17 x 22mm.

Obr. 4.2 LED7R[GHI Electronics]

4.1.2 Postup při tvorbě aplikace s potenciometrem

Nejprve se vytvoří nový projekt v prostředí Visual Studia (VS), předpokládá se, že jsou již knihovny Gadgeteeru a NETMF instalovány. Při zakládání projektu se vybere deska FEZ Spider. Poté se z lišty nástrojů vyberou moduly, které se zapojí do jednotlivých portů. Dané zapojení ve VS odpovídá fyzickému zapojení modulů k desce.



Obr. 4.3 Zpojení aplikace s potenciometrem

Ve třídě Program se nachází již defaultní metoda ProgramStared(), kterou rozšíříme o implementaci příkladu. Nejprve se vytvoří instance časovače a nastavíme ho na hodnotu 1000 ms. K němu se automaticky generuje instance s jeho delegátem a událostí, která bude zapisovat hodnotu napětí na výstup podle nastavené hodnoty.

```
GT.Timer timer = new GT.Timer(1000); // every second (1000ms)
timer.Tick += new GT.Timer.TickEventHandler(timer_Tick);
timer.Start();
```

V metodě se pomocí cyklu, který bude načítat kontrolovat hodnotu napětí na potenciometru, podle níž se rozsvítí daný počet ledek na modulu LED7R. Potenciometr pracuje s napětím 0 - 3,3V. Následně je potřeba stanovit citlivost na změnu napětí, která vyvolá změnu u rozsvícení jednotlivých diod na modulu. Výsledné řešení je vytvořeno v následujícím cyklu, který je ve smyčce pro zaznamenání změn a následné vyhodnocení.

```
while (true)
    {
        measureVolt = potentiometer.ReadPotentiometerVoltage();
        for (int i = 0; i < 7; i++)
        {
            led7r.TurnLightOff(i);
        }
        if (measureVolt > 2.5)
        {
            for (int i = 1; i < 7; i++)
            {
                led7r.TurnLightOn(i, true);
            }
        }
        else if(2.5 > measureVolt && measureVolt >2)
```

```
            {
                for(int i = 1; i< 5; i++ )
                {
                    led7r.TurnLightOn(i, true);
                }
            }
            else if (2 > measureVolt && measureVolt > 1.5)
            {
                for (int i = 1; i < 4; i++)
                {
                    led7r.TurnLightOn(i, true);
                }
            }
            else if (1.5 > measureVolt && measureVolt > 1)
            {
                for (int i = 1; i < 3; i++)
                {
                    led7r.TurnLightOn(i, true);
                }
            }
            else if (1 > measureVolt && measureVolt > 0.5)
            {
                for (int i = 1; i < 2; i++)
                {
                    led7r.TurnLightOn(i, true);
                }
            }
            else if (0.5 > measureVolt && measureVolt > 0)
            {
                for (int i = 0; i < 1; i++)
                {
                    led7r.TurnLightOn(i, true);
                }
            }
            else {
                for (int i = 0; i < 7; i++)
                {
                    led7r.TurnLightOff(i);
                }
            }
        }

    }
```

Metoda pro obsluhu časovače, kdy při zavolání metody je přečtena hodnota ve voltech na potenciometru, která se zapíše do výstupního okna ve VS.

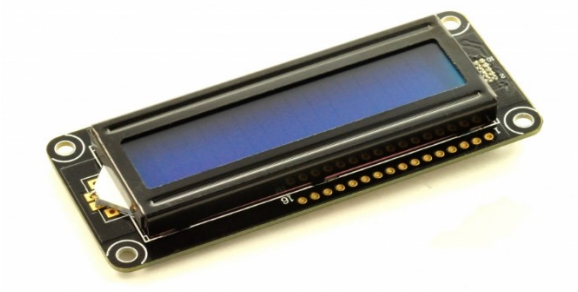
```
    void timer_Tick(GT.Timer timer)
    {
        Debug.Print("Potentiometer voltage: " + measureVolt);
    }
```

4.2 Program ke čtení kódu karet

Program má za úkol po přiložení karty k modulu čtečky karet, kartu načíst, a pokud je možné kód karty přečíst, vypíše jej na display.

4.2.1 Moduly a vývojová deska

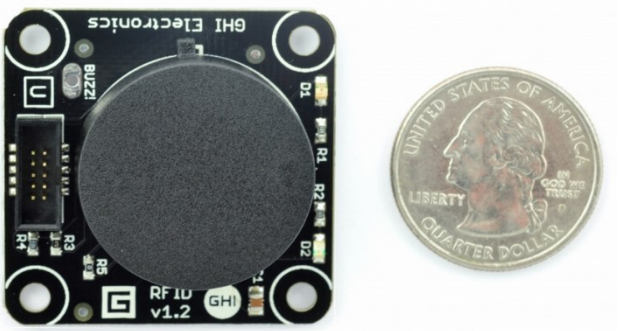
Hlavní komponentou k vytvoření aplikace je vývojová deska FEZ Raptor (popis desky se nachází v kapitole 3.3.12), ke které se následně připojí modul čtečky karet RFID, USB klient a modul Char Displaye.



Obr. 4.4 Character Display Modul

Popis externích modulů příkladu:

- Character Display modul
    - 16x2 zobrazitelných znaků obrazovky.
    - Slot typu Y
    - Napájení 3,3V s proudem 40mA nebo 5V s proudem 0mA.
    - Rozměry 87 x 27 x 15.2 mm

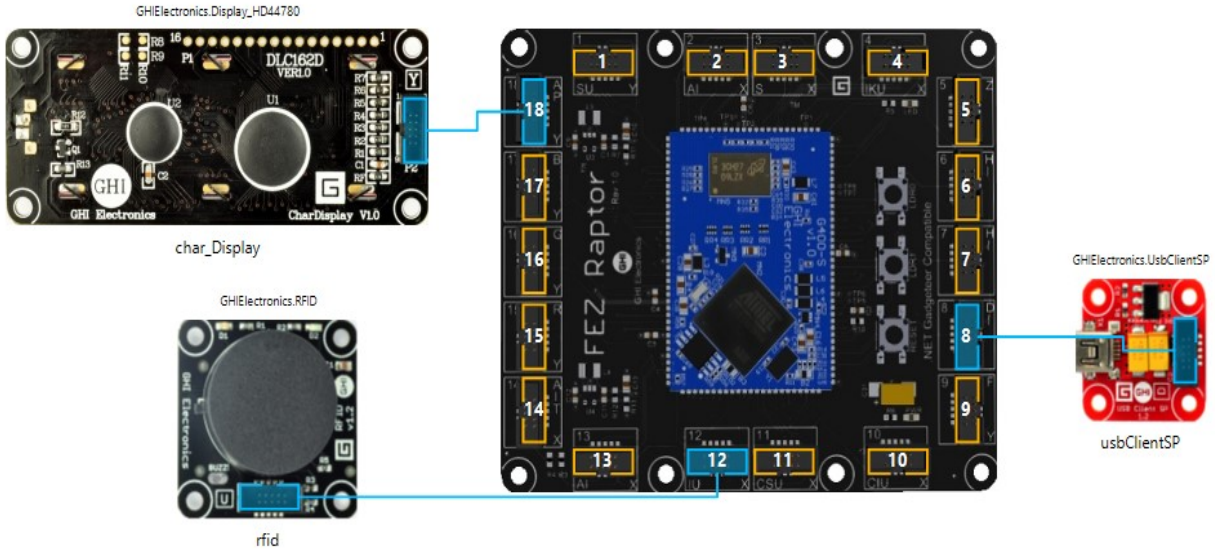


Obr. 4.5 RFID modul

- RFID modul
    - Rychlost čtení RFID znaků je 124KHz.
    - Obsahuje slot typu U.
    - Napájení 3,3V s proudem 0mA nebo 5V s proudem 1mA.
    - Rozměry 37 x 37 x 9.6 mm

4.2.2 Implementace programu

V prostředí VS se založí nový projekt. Ve druhém kroku se vybere vývojová deska FEZ Raptor a přetáhnou se k desce z toolboxu potřebné moduly, které se pak připojí k vývojové desce, viz schéma níže. Tímto krokem se automaticky nahrají potřebné knihovny a ovladače pracující se zařízeními.



Obr 4.6. Schéma zapojeni s RFID modulem

V metodě ProgramStarted() zavoláme události, které jakmile se přiblíží karta k čtečce, jsou zavolány metody `rfid_CardIDReceived` a `rfid_CardIDBadChecksum`.

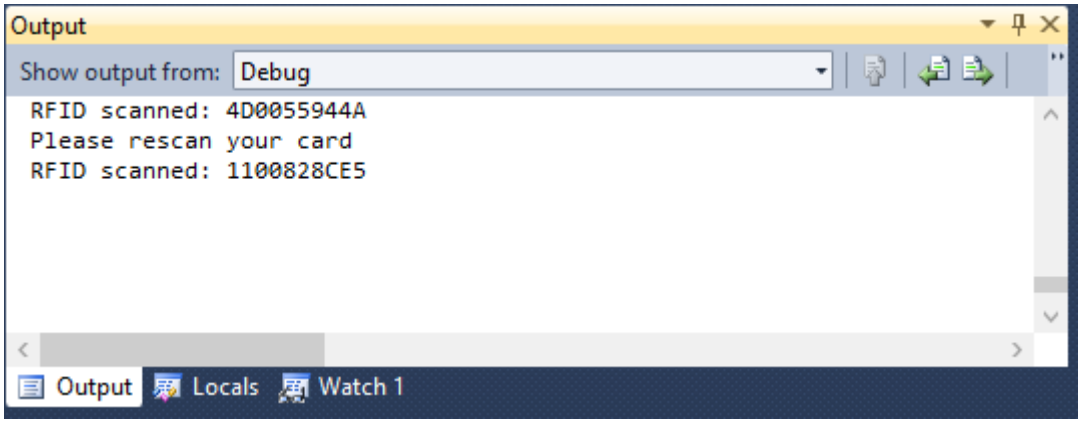
```
private void ProgramStarted()
        {
            rfid.CardIDReceived += new
RFID.CardIDReceivedEventHandler(rfid_CardIDReceived);
            rfid.CardIDBadChecksum += new
RFID.CardIDBadChecksumEventHandler(rfid_CardIDBadChecksum);
        }
```

Metoda se spustí, pokud je karta čtečkou špatně přečtena a nelze z ní dostat kód. Pak se vypíše na display a do výstupu VS upozornění, že nelze kartu přečíst.

```
void rfid_CardIDBadChecksum(RFID sender, string ID)
{
    Debug.Print("Please rescan your card");
    char_Display.Clear();
    char_Display.PrintString("Please rescan your card");
}
```

Následující kód metody vypisuje přečtený kód karty čtečkou a následně jej vypíše jak na display, tak do výstupního okna VS.

```
void rfid_CardIDReceived(RFID sender, string ID)
{
    Debug.Print("RFID scanned: " + ID);
    char_Display.Clear();
    char_Display.PrintString("RFID scanned:");
    char_Display.SetCursor(1, 0);
    char_Display.PrintString(ID);
}
```



Obr. 4.7 Výstup ve VS

5 MODULY PRO SÍŤOVOU KOMUNIKACI

Pro vybrané Gadgeteer moduly a desky jsou vytvořeny ukázkové příklady pro využití síťové komunikace a modulů zajišťující síťové připojení.

5.1 Příklad s využitím Ethernet modulu

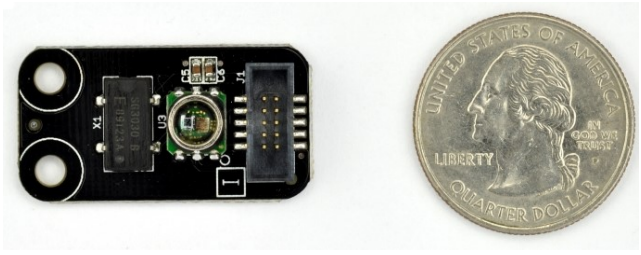
Příklad se zabývá vytvořením síťového komunikačního rozhraní s přenosem dat mezi internetovým klientem a Gadgeteer zařízením propojeným přes ethernet modul. Vhodná vývojová deska pro vypracování příkladu je deska FEZ Cerbuino Net, která má integrovaný ethernet modul. Dále obsahuje rozšiřující sloty pro připojení externích modulů. Další komponentou příkladu je senzor, který měří teplotu prostředí. Pomocí síťového připojení se budou data zobrazovat v prohlížeči připojeného počítače.

5.1.1 Hardwarové prostředky

Popis vývojové desky Cerbuino Net se nachází v kapitole 2.3.4. Externí modul připojený k Cerbuino Net na měření teploty je barometr.

Popis externích modulů příkladu:

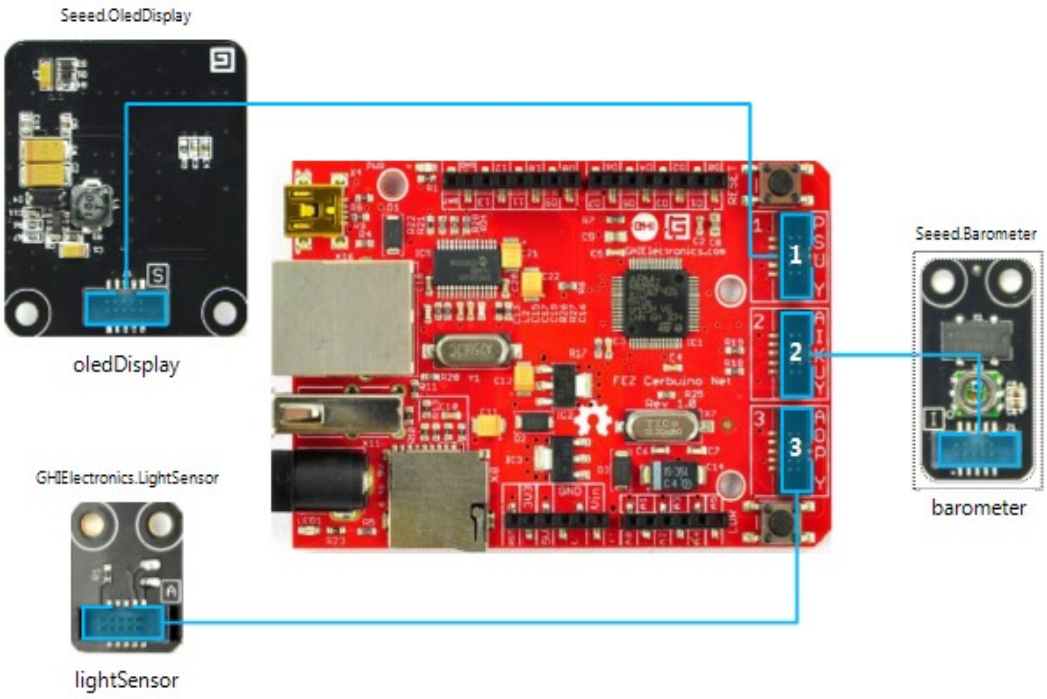
- Barometr
    - Lze využít k měření atmosférického tlaku, nadmořské výšky a teploty.
    - Typ slotu je I.
    - Napájení 3,3V s proudem < 1mA nebo 5V s proudem 0mA.
    - Provozní rozsah má v rozmezí -40°C až +85 °C
    - Velikost modulu je 17 x 37mm.



Obr. 5.1 Modul Barometr[GHI Electronics]

5.1.2 Implementace aplikace

Ve vývojovém prostředí Visual Studia se po založení nového projektu vybere deska FEZ Cerbuino Net a z toolboxu se přesunou na vizualizační plochu externí moduly, které se připojí k vývojové desce, viz schéma níže. Pro možné rozšíření příkladu jsou přidány další externí moduly.



Obr. 5.2 Schéma zapojení Cerbuino NET s moduly ve VS

Po vybrání desky Cerbuino Net ve VS, se přidají základní jmenné prostory, které se generují automaticky po založení projektu podle vybrané desky a přidaných modulů. Pro potřebu rozšíření jmenných prostorů, lze je přidat volbou Projekt – Add Reference.

Ve jmenném prostoru CerbuinoNet třídy Program se deklaruje událost na poslání hodnoty do webového rozhraní.

```
GT.Networking.WebEvent sendTeplota;
String TeplotaData;
```

V metodě ProgramStarted() je vytvořen objekt síťového rozhraní neti, ve kterém se nastaví statická IP adresa s maskou. Pomocí webového serveru, který se nastaví IP adresou a komunikačním portem.

```
void ProgramStarted()
{
    NetworkInterface neti = NetworkInterface.GetAllNetworkInterfaces()[0];
```

```
            //EthernetENC28J60 netif;
            NetworkChange.NetworkAvailabilityChanged += new
NetworkAvailabilityChangedEventHandler(NetworkChange_NetworkAvailabilityChanged);
            NetworkChange.NetworkAddressChanged += new
NetworkAddressChangedEventHandler(NetworkChange_NetworkAddressChanged);

            neti.EnableStaticIP("169.254.4.253", "255.255.0.0", "");
            Debug.Print("Program Started");

            WebServer.StartLocalServer("169.254.4.253", 80);
```

Nastavení obsluhy měřících událostí v časovém rozpětí 2000ms a deklarace modulu barometr pro měření teploty, která je poslána pomocí webové události sendTeplota klientovi od webového serveru.

```
            GT.Timer timer = new GT.Timer(2000);
            barometer.MeasurementComplete += new
Barometer.MeasurementCompleteEventHandler(barometer_MeasurementComplete);
            barometer.StartContinuousMeasurements();

            sendTeplota = WebServer.SetupWebEvent("temperature");
            sendTeplota.WebEventReceived += new
WebEvent.ReceivedWebEventHandler(sendTeplota_WebEventReceived);

        }
```

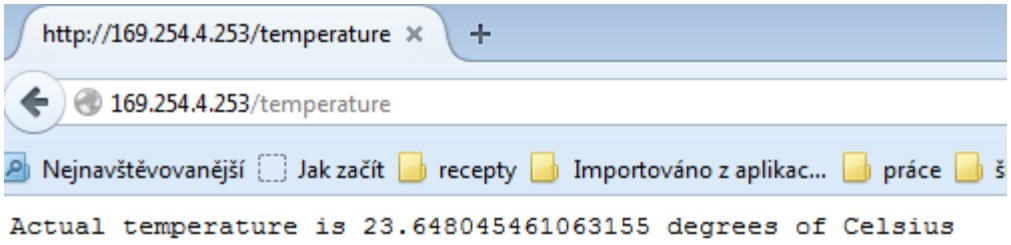
Metoda slouží k předání dat v daném formátu aktuální teploty snímané barometrem. Řetězec je uložen do proměnné TeplotaData. Dále je pro kontrolu vytvořen výstup do výstupu ve VS, která se zapíše po spuštění programu v režimu Debug.

```
        void barometer_MeasurementComplete(Barometer sender, Barometer.SensorData
sensorData)
        {
            Debug.Print("Temperature: " + sensorData.Temperature + " °C. Presurre:
" + sensorData.Pressure);
            TeplotaData = "Actual temperature is " +
sensorData.Temperature.ToString() + " °C";
        }
```

Implementace události, kdy je v předchozí metodě vytvořen řetězec, který se po vygenerování pošle server webovému prohlížeči jako odpověď klientovi od serveru.

```
        void sendTeplota_WebEventReceived(string path, WebServer.HttpMethod
method, Responder responder)
        { responder.Respond(TeplotaData); }
```

Zobrazení aktuální hodnoty na webovém prohlížeči s nastavenou adresou.



Obr. 5.3 Výpis aktuální hodnoty na webovém prohlížeči

5.2 Ukázkový příklad s využitím WiFi modulu

Připojením externího WiFi modulu k libovolné vývojové desce zjistěte dostupné SSID sítě v okolí zařízení. Ke každé dostupné síti se následně vypíšou s parametry z každé sítě.

5.2.1 Hardwarové komponenty příkladu

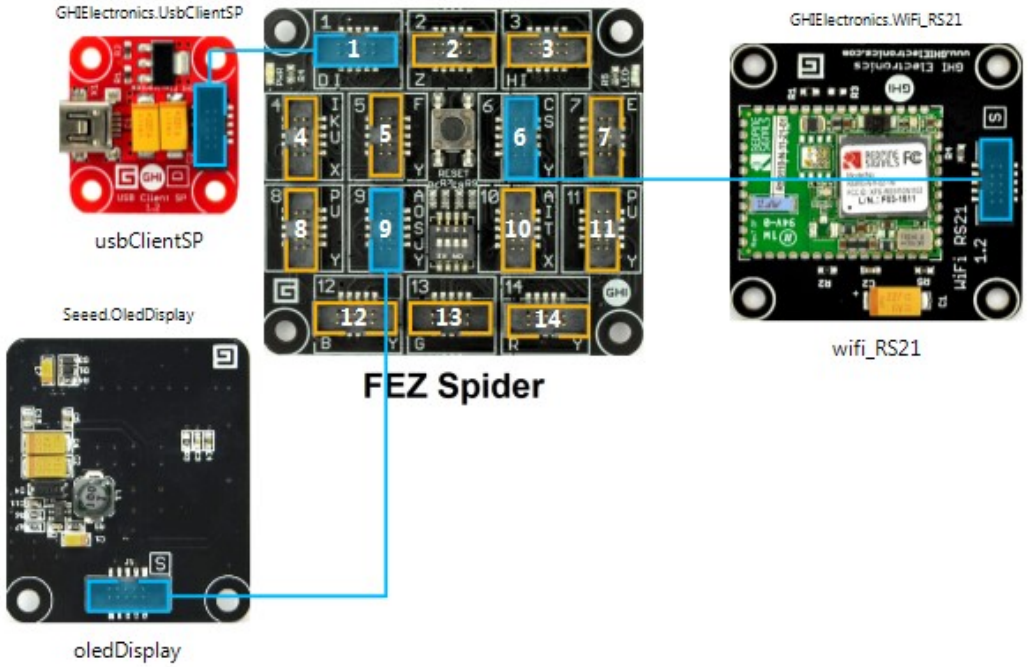
Příklad se skládá z vývojové desky FEZ Spider, která je popsaná v kapitole 2.3.1, USB Client SP 1.3, který je popsán v kapitole 3.1.1 a externího WiFi modulu typu RS21.

- Popis modulu WiFi RS21
  - Určený pro vývojovou desku FEZ Spider
  - Typ slotu je S.
  - Napájení 3,3V s proudem 40mA nebo 5V s proudem 0mA.
  - Velikost modulu je 42 x 42mm.

Obr. 5.4 WiFi RS21 Modul

5.2.2 Implementace příkladu

Ve vývojovém prostředí VS se založí nový projekt s názvem SpiderWifi. Následně se vybere vývojová deska FEZ Spider, ke které se připojí externí moduly. Poté se k těmto modulům vygenerují jednotlivé jmenné prostory pro využití daných knihoven potřebných k využití již implementovaných metod k jednotlivým zařízením.



Obr. 5.5 Schéma zapojení s WiFi modulem ve VS

```
using System.Collections;
using System.Threading;
using Microsoft.SPOT;
using Gadgeteer.Networking;
using GT = Gadgeteer;
using GTM = Gadgeteer.Modules;
using Gadgeteer.Modules.GHIElectronics;
using Gadgeteer.Modules.Seeed;
```

Ve třídě Program, který je obsažen v projektu SpiderWifi se nachází metoda ProgramStarted() v níž se volají jednotlivé metody pro zjištění okolních WiFi sítí.

```
Debug.Print("running");
```

Metoda UseDHCP() používá nastavení DHCP routru pro zjištění informací.

```
wifi_RS21.UseDHCP();
```

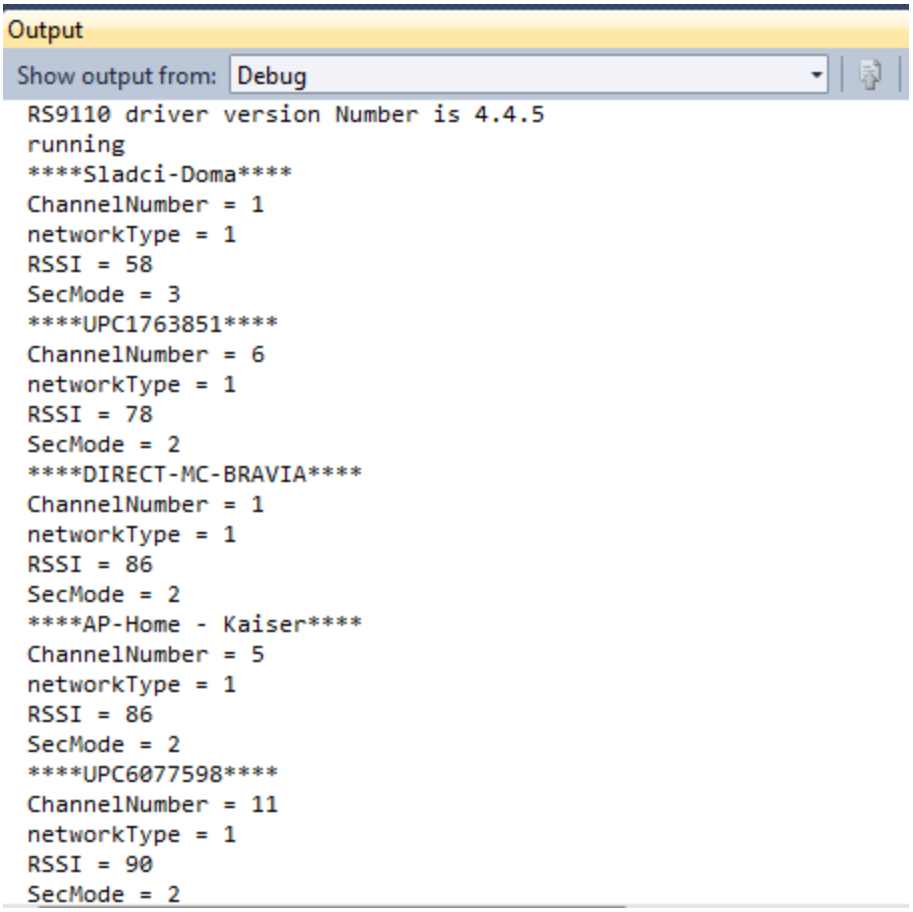
Metoda Scan() skenuje okolí zařízení. Hledá SSID dostupných WiFi sítí, ze kterých získá informace o nalezených sítích a ty uloží do pole scanResults.

```
            GHI.Premium.Net.WiFiNetworkInfo[] scanResults =
wifi_RS21.Interface.Scan();
```

Pomocí následujícího cyklu jsou nalezené sítě postupně vypsány v daném formátu do výstupu ve VS při spuštění Debug procesu.

```
            foreach (GHI.Premium.Net.WiFiNetworkInfo result in scanResults)
            {
                Debug.Print("****" + result.SSID + "****");
                Debug.Print("ChannelNumber = " + result.ChannelNumber);
                Debug.Print("networkType = " + result.networkType);
                Debug.Print("RSSI = " + result.RSSI);
                Debug.Print("SecMode = " + result.SecMode);
            }
```

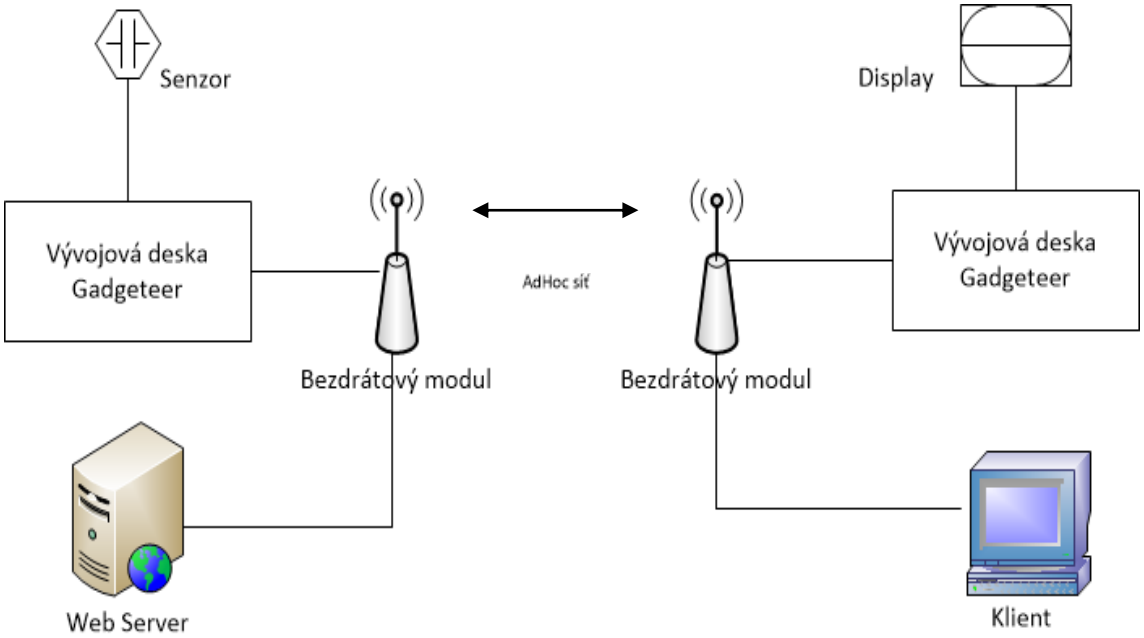
Popis nalezených sítí ve VS oknu Output pro výstup aplikace spuštěné v režimu Debug. Channel number značí číslo kanálu, na kterém se WiFi síť nalézá. Network Type určuje, zda jde o připojení typu AdHoc při čísle 0 nebo přístupový bod při čísle 1. RSSI je indikátor síly signálu a SecMode určuje zabezpečení sítě, kdy číslo 2 vyjadřuje zabezpečení typu WPA2 a číslo 3 je WEP.



Obr. 5.6 Výstup aplikace: výpis WiFi sítí

5.3 Wifi síť

Aplikace vytvoří WiFi síť založenou na standardu IEEE 802.11b/g. V rámci vytvořené sítě komunikují dvě desky. První vývojová deska pomocí WiFi modulu síť vytváří. Má spuštěný web server a zároveň sbírá data ze senzoru, které jsou dostupné přes web server. Druhá deska připojená do sítě a obsahuje klienta, který se dotazuje IP adresou serveru na data uložená na web serveru, kde se posílají naměřená data ze senzoru. Při dotazu http klienta na web server jsou naměřená data z první desky poslána a následně zobrazena na displeji a ve výstupním okně debuggu. Komunikace probíhá v aplikační vrstvě, kde jsou pro přenos dat využívány funkce http protokolu z knihoven bezdrátového WiFi modulu.



Obr 5.7 Schéma sítě

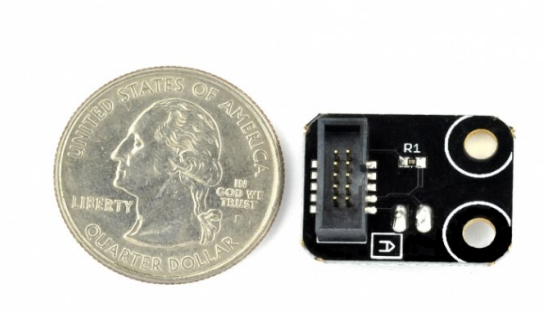
5.3.1 Hardwarové komponenty

Základ tvoří dvě vývojové desky. První vývojová deska je FEZ Spider (kapitola 3.3.1), na které se nachází server obsahující naměřená data a tvořící síť. Druhá deska je FEZ Raptor (kapitola 3.3.2), kde se nachází http klient, ze kterého jsou přijímané data zobrazena.

Měřená data jsou teplota z připojeného modulu barometr (kapitola 5.1.1) a intenzitu světla okolí pomocí modulu LightSence měřenou v procentech. Pro vytvoření WiFi sítě mají vývojové desky připojené WiFi moduly WiFi RS21(kapitola 5.2.1). Přijímaná data se zobrazují na modulu CharDisplay (kapitola 4.2.1).

Popis použitých externích modulů aplikace:

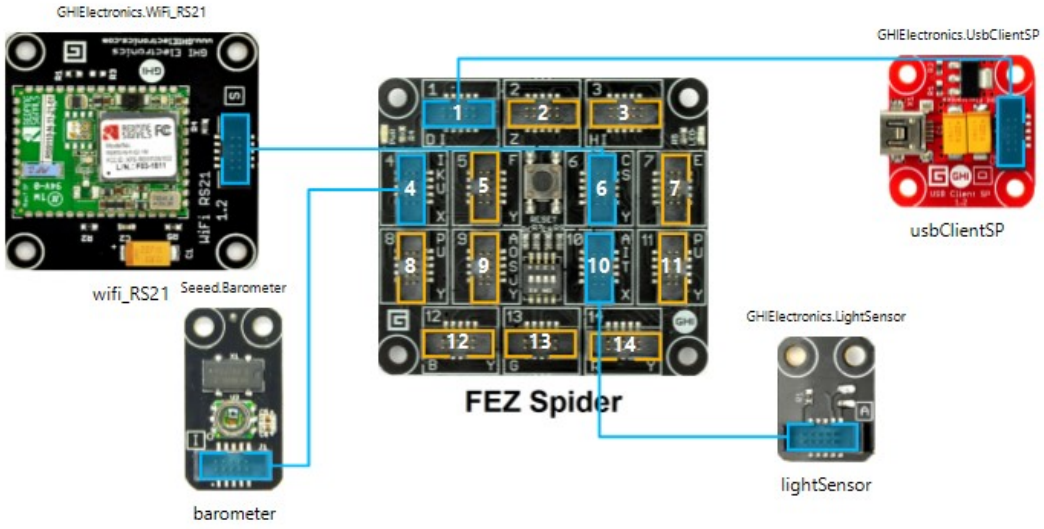
- Modul LightSense
    - Měří intenzitu světla
    - Slot typu A.
    - Napájení 3,3V s proudem 40mA nebo 5V s proudem 0mA.
    - Velikost modulu je 17 x 22 mm



Obr. 5.8 LightSense Modul

5.3.2 Implementace programu

WiFi síť se vytvoří s použitím WiFi modulu. Pomocí funkcí síťových knihoven se konfiguruje síť. Pro správnou funkci sítě je potřebné nastavit rozhraní, informace o síti, IP adresa sítě, ať už staticky nebo pomocí DHCP serveru dynamicky. Pro přeposílání dat se vytvoří ad-hoc komunikační síťové rozhraní pro dočasné propojení mezi zařízeními, které slouží ke sdílení internetu, či sdílení souborů. U konfigurace ad-hoc připojení je zapotřebí nastavit SSID, vybrat jeden z dostupných typů šifrování, heslo sítě a číslo komunikačního kanálu.



Obr. 5.9 Schéma zapojení desky se senzory

Aplikaci tvoří dva programové celky, jelikož každá vývojová deska obsahuje vlastní program. Podle připojených modulů se vytvořily samostatné programy pro:

A. Vytvoření sítě a sběru dat ze sensorů

Schéma zapojení vývojové desky s jejím bezdrátovým modulem a senzorickými moduly ve VS zobrazuje obr. 5.9.

V následujícím kódu je nastavení GT.Timer časovače pro spouštění události, která zajistí čtení naměřených hodnot z barometru.

```
            GT.Timer timer = new GT.Timer(2000);
            barometer.MeasurementComplete +=new
Barometer.MeasurementCompleteEventHandler(barometer_MeasurementComplete);
            barometer.StartContinuousMeasurements();
```

Dále se musí ověřit, zda bezdrátové rozhraní modulu je otevřeno, pak se může nastavit potřebná konfigurace sítě a následně lze vytvořit připojení typu ad-hoc mezi vývojovými deskami sloužící ke sdílení dat. Nesmí se však zapomenout na nastavení události `Interface.WirelessConnectivityChanged`, která zavolá metodu web serveru sloužící ke konfiguraci.

```
            GHI.Premium.Net.WiFiNetworkInfo info = new
GHI.Premium.Net.WiFiNetworkInfo();
            info.SSID = "MyNetwork";

            if (!wifi_RS21.Interface.IsOpen)
            {
                wifi_RS21.Interface.Open();
            }
            wifi_RS21.UseStaticIP("169.254.4.253", "255.255.0.0", "");
```

```
            wifi_RS21.Interface.StartAdHocHost("MyNetwork",
GHI.Premium.Net.SecurityMode.Open, "12345", 1);

            wifi_RS21.Interface.WirelessConnectivityChanged += new
WiFiRS9110.WirelessConnectivityChangedEventHandler(Interface_WirelessConnectivityC
hanged);
```

Aby bylo možné posílat data přes již nastavenou síť, je zapotřebí inicializovat web server, který je spouštěn, jako událost viz text níže. Následuje metoda `Interface_WirelessConnectivityChanged`, která nastaví IP adresu serveru a přístupový port. Komunikace se serverem se vytvoří, jakmile se na nastavenou IP adresu a port klientem pošle požadavek.

```
            getSensorPersentage = WebServer.SetupWebEvent("SensorPersentage");
            getSensorPersentage.WebEventReceived += new
WebEvent.ReceivedWebEventHandler(getSensorPersentage_WebEventReceived);

            void Interface_WirelessConnectivityChanged(object sender,
WiFiRS9110.WirelessConnectivityEventArgs e)
        {
            if (e.IsConnected)
            {
                WebServer.StartLocalServer("169.254.4.253", 8080);
            }

        }
```

Následující kód je metoda, která sbírá data ze senzorů a ukládá je do proměnných, které jsou ve formátu řetězce pro výpis jak na obrazovku nebo na výstup VS, či prohlížeče.

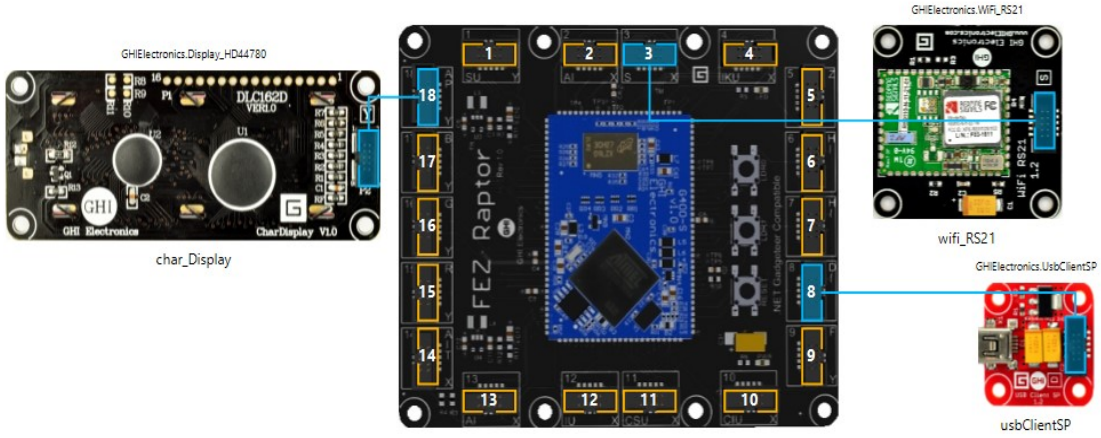
```
            void barometer_MeasurementComplete(Barometer sender,
Barometer.SensorData sensorData)
            {
                anyDoubleVariable = (int)lightSensor.ReadLightSensorPercentage();
                sensorPersentage = "Light is " + anyDoubleVariable.ToString() +
"%\n" + "Temperature is " + sensorData.Temperature.ToString() + " °C";
                Debug.Print(sensorPersentage);

            }
```

Metoda web serveru, která se spustí, jakmile se zavolá událost `getSensorPersentage.WebEventReceived`, která má za úkol poslat přes web server data ze senzoru uložené v proměnné `sensorPersentage`. Vše se provede po vyslání requestu od klienta umístěného ve druhém zařízení komunikace.

```
       private void getSensorPersentage_WebEventReceived(string path,
WebServer.HttpMethod method, Responder responder)
       {
           responder.Respond(sensorPersentage);
           Debug.Print("komunikace");
       }
```

B. Vytvoření klienta a zobrazení dat

Vývojová deska obsahuje bezdrátový modul pro vytvoření WiFi připojení s klientem a modul obrazovky k zobrazení přijatých dat. Schéma zapojení se nachází na obr. 5.10.



Obr. 5.10 Schéma zapojení desky s obrazovkou

Program nejprve zjišťuje dostupnost sítí v jeho okolí, o kterých zjistí informace, jako je název sítě, číslo kanálu, typ sítě a RSSI. Zjištěné informace se uloží do pole typu `WiFiNetworkInfo` a následně vypíší do výstupního okna ve VS.

```
            GHI.Premium.Net.WiFiNetworkInfo[] info = null;
            GHI.Premium.Net.WiFiNetworkInfo[] scanResults =
wifi_RS21.Interface.Scan();

            foreach (GHI.Premium.Net.WiFiNetworkInfo result in scanResults)
            {
                Debug.Print("****" + result.SSID + "****");
                Debug.Print("ChannelNumber = " + result.ChannelNumber);
                Debug.Print("networkType = " + result.networkType);
                Debug.Print("RSSI = " + result.RSSI);
                Debug.Print("SecMode = " + result.SecMode);

            }
```

Poté se podle SSID se vyhledá síť a ověří se její dostupnost. Pokud se síť jeví jako dostupná, tak se na ní WiFi modul připojí.

```
            info = wifi_RS21.Interface.Scan("MyNetwork");
            if (info != null)
            {
                wifi_RS21.Interface.Join(info[0], "12345");
                Debug.Print("Network joined");
```

Po úspěšném připojení do sítě se provede konfigurace tak, že se staticky nastaví IP adresa a maska pro bezdrátovou komunikaci. Pro ověření úspěšnosti nastavení IP adresy se vypíše adresa a maska sítě do výstupu okna ve VS.

```
            wifi_RS21.UseStaticIP("169.254.4.254", "255.255.255.0", "");

            Debug.Print("IP Address: " + wifi_RS21.NetworkSettings.IPAddress);
            Debug.Print(wifi_RS21.Interface.NetworkInterface.GatewayAddress);
```

Nakonec se pro komunikaci s web serverem vytvoří událost web klient, který bude serveru posílat http požadavek, a tím dostane požadovanou odpověď. Pro navázání komunikace se serverem je nutné uvézt GET požadavek od klienta, kde se uvádí IP adresa serveru, port a popřípadě adresář, ve kterém se případné data nachází.

```
            Gadgeteer.Networking.HttpRequest wc =
        WebClient.GetFromWeb("http://169.254.4.253:8080/SensorPersentage");
            wc.ResponseReceived += new
        HttpRequest.ResponseHandler(wc_ResponseReceived);

    }
```

Metoda události má za úkol přijaté data od serveru zpracovat a vypsat je na display i do výstupního okna ve VS.

```
void wc_ResponseReceived(HttpRequest sender, HttpResponse response)
{
    string text = response.Text;
    Debug.Print(text);
    char_Display.Clear();
    char_Display.PrintString(text);
}
```

6 ROBOTIKA

Pro robotickou sadu FEZ Cerbot jsou vytvořeny ukázkové aplikace pro řízení a regulaci s použitím externích a integrovaných modulů sady Cerbot.

6.1 Aplikace s využitím integrovaných modulů FEZ Cerbot

Pro využití integrovaných modulů je vytvořena aplikace, která kopíruje trajektorii černé čáry a řídí tak svůj pohyb po ní. Základem je použitá sada FEZ Cerbot, která je popsána v kapitole 3.3.5.

6.1.1 Hardwarové komponenty projektu

Hardwarové komponenty pro vytvoření příkladu jsou součástí balíčku FEZ Cerbot. Ke sledování čáry má deska Cerbot integrované dva reflexní senzory, které jsou umístěny na přední části desky. Pohyb desky zajišťují motorky umístěné na bočních stranách spolu s koly a pro vyrovnání desky je přidáno kolečko doprostřed přední části desky, viz Obr. 3..

Obr. 6.1 Schéma desky Cerbot ve VS

6.1.2 Implementace aplikace

V prostředí Visual Studia se vytvoří nový projekt. Při zakládání projektu se vybere deska FEZ Cerbot. Vzhledem k tomu, že na desce jsou potřebné moduly integrované, avšak pro rozšíření úlohy lze připojit externí moduly, jako jsou gyroskop, akcelerometr. Schéma desky ve VS lze vidět na obr. 6.1.

Pohyb Cerbot je událostně řízen, a to tak, že se pro změnu směru vyvolá událost podle zjištěných hodnot na reflexním snímači.

Po vybrání desky Cerbot ve VS, se automaticky přidají základní jmenné prostory potřebné pro práci s deskou a jejími integrovanými moduly.

```
using System;
using System.Collections;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Controls;
using Microsoft.SPOT.Presentation.Media;
using Microsoft.SPOT.Presentation.Shapes;
using Microsoft.SPOT.Touch;

using Gadgeteer.Modules;
using Gadgeteer.Networking;
using GT = Gadgeteer;
using Gadgeteer.Modules.GHIElectronics;
```

Ve jmenném prostoru CerbotProject se nachází třída Program, v níž je implementována aplikace. Pohyb desky Cerbot je řešen událostně, což vede k vytvoření delegátu a následně daných událostí, které určují nastavení motorů a tím řídí pohyb.

```
public delegate void MyEdgeDetectDelegate();
public event MyEdgeDetectDelegate rightEdge;
public event MyEdgeDetectDelegate leftEdge;
public event MyEdgeDetectDelegate bothEdge;
public event MyEdgeDetectDelegate nonEdge;
```

Pro nastavení desky nebo komunikaci s moduly desky je potřeba vytvořit instanci desky:

```
CerbotController cerbotController = new CerbotController();
```

Následně se nastaví konstantní rychlost jakou má se Cerbot pohybovat. Rychlost lze nastavit od 0 do 100% otáček motorů.

```
const int RUN_SPEED = 60;
const int REVERSE_SPEED = -60;
```

V metodě ProgramStarted jsou deklarovány události a nastavení směru otáčení motorků. Směr motorků se liší, z důvodu zapojení jednoho motorku inverzně, to však lze vyřešit programově, kdy se otočí směr motoru.

```
        void ProgramStarted()
        {
            cerbotController.SetMotorInversion(false, true);

            this.bothEdge += new MyEdgeDetectDelegate(Program_bothEdge);
            this.leftEdge += new MyEdgeDetectDelegate(Program_leftEdge);
            this.rightEdge += new MyEdgeDetectDelegate(Program_rightEdge);
            this.nonEdge += new MyEdgeDetectDelegate(Program_nonEdge);
```

Pro oddělení řízení motorů od ostatních metod je vytvořené samostatné vlákno, které zajistí samostatný běh řízení bez přerušení.

```
new Thread(DoEdgeDetect).Start();
        }
```

V následujícím kódu jsou vytvořeny těla událostí, ve kterých se nachází nastavení rychlosti motorů Cerbotu a podle typu vyvolané události je řízen směr otáčení motorů a jejich rychlost.

```
        void Program_nonEdge()
        {
            cerbotController.SetMotorSpeed(RUN_SPEED, RUN_SPEED);
        }

        void Program_rightEdge()
        {
            cerbotController.SetMotorSpeed(RUN_SPEED, REVERSE_SPEED);
        }

        void Program_leftEdge()
        {

            cerbotController.SetMotorSpeed( REVERSE_SPEED, RUN_SPEED);
        }

        void Program_bothEdge()
        {
            cerbotController.SetMotorSpeed(REVERSE_SPEED, REVERSE_SPEED);
        }
```

V metodě DoEdgeDetect() se získají hodnoty z pravého a levého reflexního senzoru. V nekonečném cyklu se čtou hodnoty z levého i pravého senzoru, které jsou ukládány do deklarovaných proměnných.

Hodnoty ze senzoru se porovnají s hodnotou mezní konstanty. Tím se vyhodnotí směr jízdy Cerbotu, podle kterého je následně vyvolána odpovídající událost.

```
        void DoEdgeDetect()
         {
             const double SENSOR_THRESHOLD = 50;
             double intensReflexL =0;
             double intensReflexR = 0;

  while (true)
             {
                 intensReflexL =
cerbotController.GetReflectiveReading(CerbotController.ReflectiveSensors.Left);
                 intensReflexR =
cerbotController.GetReflectiveReading(CerbotController.ReflectiveSensors.Right);
                 if (intensReflexL >SENSOR_THRESHOLD && intensReflexR >
SENSOR_THRESHOLD)
                 {
                     this.nonEdge();
                 }
                 else if (intensReflexL > SENSOR_THRESHOLD)
                 {
                     this.rightEdge();
                 }
                 else if (intensReflexR > SENSOR_THRESHOLD)
                 {

                     this.leftEdge();
                 }
                 else
                 {
                     this.bothEdge();
                 }
             }
```

6.2 Ukázkový příklad balancování s FEZ Cerbot

Příklad se inspiruje projektem vývojářské komunity GHI Electronics. Součástí projektu je FEZ Cerbot s externími moduly Gyroskopu a Akcelerometru. Aplikace dokáže dosáhnout balancování desky a vyrovnávat drobné vychýlení působící na desku. Pro výpočet rychlosti motorků k balancování Cerbotu je použitý PID regulátor spolu s naměřenými daty z akcelerometru a gyroskopu. Z naměřených dat senzorů se vypočítávají odchylky od požadovaného úhlu a přírůstkový PSD regulátor má za úkol tuto odchylku odstranit.

$$u(kT) - u[(k-1)T] = K_p\{e(kT) - e[(k-1)T] + \frac{T}{T_I}K_I\, e(kT) +$$

$$+ \frac{T_D}{T}\{e(kT) - 2e[(k-1)T] + 2e[(k-2)T]\}\} \quad (1)$$

Úhel náklonu je tvořen z výstupních hodnot z akcelerometru a gyroskopu. Aktuální hodnotu úhlu upravujeme pomocí dolnopropustního filtru, který vychází z ověřených předchozích hodnot, ke kterým se přičtou nově výstupní hodnoty senzorů, avšak s minimální váhou.

Pro vyhodnocení úhlu zařízení je použitý následující vztah vycházející ze součtu filtrovaných hodnot akcelerometru a gyroskopu s předchozím úhlem zařízení:

$$\alpha_{k+1} = 0{,}98 * (\alpha_k + gyro * dt) + 0{,}02 * acc \quad (2)$$

6.2.1 Komponenty příkladu

Základ tvoří vývojová sada FEZ Cerbot (popis se nachází v kapitole 3.3.5) a k ní připojené externí moduly gyroskopu, akcelerometru pro zjištění úhlu náklonu desky a směr pohybu, dále je připojen modul tlačítka pro spuštění regulace.

Popis přidaných externích modulů:

- Modul Gyroskop
    - Tři osy MEMS senzor pohybu
    - Měří úhlovou rychlost
    - Komunikační rozhraní I2C
    - Slot typu I.

- Napájení 3,3V s proudem 40mA nebo 5V s proudem 0mA.
- Velikost modulu je 32 x 17 x 7,15 mm

Obr. 6.2 Gyroskop modul

- Modul Akcelerometr
  - Dvou osy senzor měřící zrychlení
  - Komunikační rozhraní I2C
  - Slot typu I.
  - Napájení 3,3V s proudem 40mA nebo 5V s proudem 0mA.
  - Velikost modulu je 32 x 17 x 7,15 mm

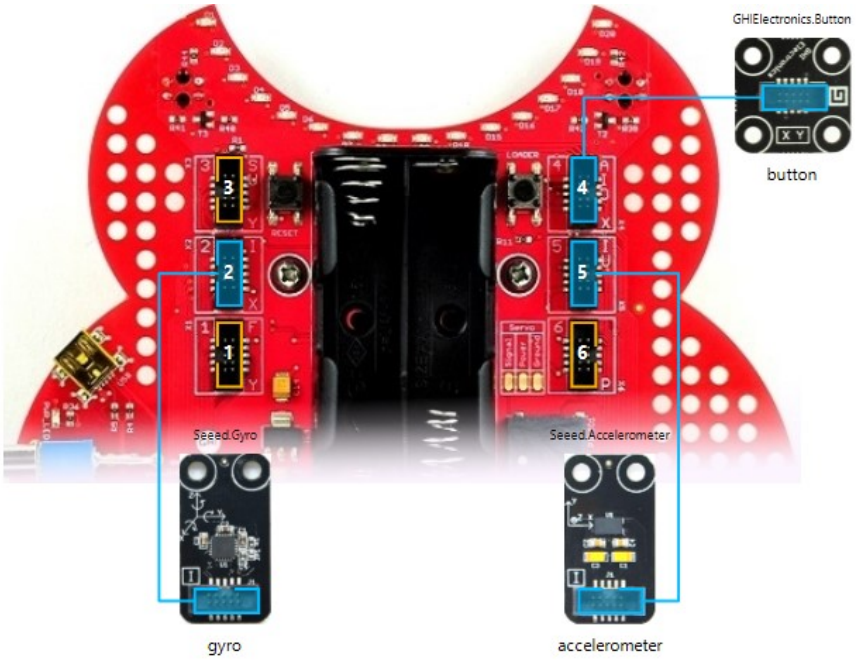
Obr. 6.3 Akcelerometr modul

- Button modul
  - Součástí je LED dioda
  - Slot typu X,Y.
  - Napájení 3,3V s proudem 10mA nebo 5V s proudem 0mA.
  - Velikost modulu je 22 x 22 x 15 mm

Obr. 6.4 Modul tlačítka

6.2.2 Implementace aplikace

Ve vývojovém prostředí se vytvoří projekt tak, jak je to popsáno v kapitole 6.1.2, kde se navíc přidají externí moduly akcelerometru, gyroskop a tlačítko do schématu, ve kterém je zobrazeno zapojení jednotlivých modulů, viz obr. 6.6.



Obr. 6.5 Zapojení Cerbotu a modulů ve VS

V programu se nejprve kalibruje akcelerometr s gyroskopem, kdy se vytváří tzv. offset, což značí seřízení senzorů v klidovém výchozím stavu, proto je nutné, aby se Cerbot nacházel ve vodorovné poloze. Po kalibraci je potřeba Cerbota nastavit do svislé polohy, ve které se zmáčkne tlačítko spouštějící regulaci balancování ve svislé poloze.

Požadovaná hodnota úhlu ve svislé poloze je 270°. Aktuální úhel Cerbotu se získává z hodnot akcelerometru a gyroskopu, viz vztah (2). Odchylka (_error) se získá rozdílem požadované hodnoty od aktuální. Pro regulaci se uchovávají hodnoty předchozích odchylek o dvě hodnoty zpátky, což znamená předchozí hodnota odchylky (_prevError) a předchozí hodnota předchozí odchylky (_prevError2).

Balancování Cerbotu ve svislém směru tvoří přírůstkový regulátor PSD. Výpočet regulátoru vychází ze vztahu pro výpočet přírůstkového regulátoru, viz vztah (1).

```
output = output + Kp * (_error - _prevError + (Ki * _error)/Kp + Kd * (_error - 2
* _prevError + _prevError2)/Kp);
```

V předchozí kód znázorňuje již upravený vztah pro výpočet regulátoru použitý v aplikaci, kde Kp, Ki, Kd jsou koeficienty regulátoru získané z experimentálního nastavení pro nejstabilnější průběh. Výpočet akční veličiny pro rychlost otáčení motorků lze nastavit v rozmezí od 0 do 100 %, která vychází z výstupní hodnoty regulátoru (output) a je násobena konstantou pro rychlost.

```
_speed = (int)(MotorSpeedFactor * output + 0.5);
```

ZÁVĚR

Cílem práce bylo představit platformu .NET Gadgeteer jako technologii pro tvorbu malých elektronických projektů a embedded systémů. .NET Gadgeteer platforma je postavena na technologii .NET Micro Framework jejíž architekturu jsem v práci popsala a technologii FEZ hardwarových zařízení.

Představila jsem hardwarovou část systému Gadgeteer, kde jsem popsala základní řídicí moduly vývojových desek a dostupné externí moduly. Vybrala jsem vývojové desky podle jejich dostupnosti a možnosti využití, které jsem následně popsala. Dále jsem uvedla kategorie dostupných modulů pro rozšíření vývojových desek a tvorbu menších elektronických projektů.

V aplikacích jsem navrhla a implementovala vlastní příklady pro vybrané vývojové desky FEZ Spider, FEZ Raptor, FEZ Cerbuino Net, FEZ Cerbot s využitím vybraných senzorických, komunikačních, zobrazovacích externích modulů a modulů uživatelských vstupů.

Před implementací navržených aplikací bylo nutné nakonfigurovat jednotlivé vývojové desky aktuálním firmwarem spolupracujícím s vývojovým prostředím a dostupnými knihovnami .NET Gadgeteer Core a SDK nástroji NETMF. Při tvorbě projektů jsem využívala ladící debug nástroj vývojového prostředí VS pro .NET, který umožňuje sledovat v průběhu aplikace proměnné a chování jednotlivých objektů.

V úvodní části věnované praktickým aplikacím jsou představeny funkcionality senzorických modulů a zobrazovacích jednotek. Následující část se zabývá síťovými moduly. V implementovaných aplikacích jsem využila vývojovou desku s integrovaným Ethernet rozhraním, které je součástí vývojové desky FEZ Cerbuino Net. K desce je připojen modul barometru, který snímá teplotu a tlak okolí. Získaná data z barometru jsem vypsala do webového prohlížeče počítače, ke kterému bylo zařízení pomocí Ethernet rozhraní připojeno. V následujících aplikacích jsem využila externí WiFi moduly pro bezdrátovou síť. Nejprve jsem představila aplikaci s modulem, který skenuje blízké okolí a hledá SSID okolních WiFi sítí a získá popis sítě, jako je typ zabezpečení, síla signálu, typ připojení a číslo kanálu. Navazující projekt využívá předchozí aplikaci, a dále pak vytvoří komunikační kanál s použitím ad-hoc. Projekt implementuje dvě aplikace pro dvě

vývojové desky. Jedna deska FEZ Spider síť vytvoří a poskytuje přes web server naměřená data, druhá deska FEZ Raptor data pomocí http klienta přijímá a zobrazuje na display.

Poslední část práce tvoří oblast robotiky se sadou robotického vozítka FEZ Cerbot. Byly vytvořeny dva projekty. V prvním jsem použila integrované moduly vývojové desky, se kterými jsem vytvořila událostně řízenou aplikaci pro sledování čáry. Vozítko sleduje čáru, a to pomocí dvou reflexních senzorů, podle kterých se následně řídí pohyb motorků. Ve druhém projektu jsem implementovala aplikaci s rozšiřujícími moduly gyroskopu a akcelerometru, které využijeme pro balancující vozítko typu segway ve svislé poloze s použitím přírůstkového PSD regulátoru.

Součástí diplomové práce je příloha se všemi zdrojovými kódy jednotlivých projektů a video ukázka vytvořených aplikací. Tyto přílohy jsou dostupné na přiloženém CD.

SEZNAM POUŽITÉ LITERATURY

GHI ELECTRONICS LLC, *GHI Tutorials* [online]. [cit. 2014-07-04]. Dostupné z: www.ghielectronics.com/docs/37/netmf-and-gadgeteer-tutorial-index

KAČMAŘ, D. *Programujeme .NET aplikace ve Visual Studiu .NET*. Computer Press 2001, ISBN 80-7226-569-5

KUHNER, J. *Expert .NET Micro Framework*. Apress, 2008. ISBN 978-1-59059-973-0. Dostupné z: http://it-ebooks.info/book/2053/

MALIN, J. a LIMING S. *Professional's Guide To .NET Micro Framework Application Development.* Annabook, 2012, ISBN-13 978-0-9842801-9-3

MONK, S. *Getting Started with .NET Gadgeteer*. O´Reilly Media, 2012. ISBN 978-1-449-32823-8.

MICROSOFT. *Microsoft Developer Network*. [online]. [cit. 2014-03-19]. Dostupné z: http://msdn.microsoft.com/

MICROSOFT, *Micro .NET Framework*. [online]. [cit. 2014-03-04]. Dostupné z: http://www.netmf.com/

MICROSOFT. *Microsoft Visual Studio*. [online]. [cit. 2014-07-04]. Dostupné z: www.microsoft.com/cze/msdn/vstudio/

SYTECH DESIGN LTD, *Sytech Designs web portal*, [online]. [cit. 2015-01-10]. Dostupné z: http://www.sytechdesigns. com/micro_framework.htm

THOMSON, D. a MILES R. *Embedded Programming with the Microsoft .NET Micro Framework*. 4. vyd. Microsoft Press, 2007. ISBN 978-0735623651.

VÍTEČKOVÁ, M. a VÍTEČEK A. *Základy automatické regulace.* VŠB – Technická univerzita Ostrava, 2006. ISBN 978-80-248-1924-2

NOSKIEVIČ,P. *Modelování a identifikace systémů.* Montanex, Ostrava, 1999

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

- CD
    - Sladka_Karla_text_DP
    - Zaznam_prace
    - Aplikace_zdrojove_kody